Group_inflation_comparison

February 11, 2025

[6]:
```
from inflation_analysis import grouping, calculate_price_indexes
from tabulate import tabulate
from tqdm import tqdm
import matplotlib.pyplot as plt
import pandas as pd


# Parameters
start_year = 2019
end_year = 2022
data_folder="/Users/roykisluk/Downloads/Consumer_Expenditure_Survey/"
top_n = 10
base_year = start_year
years=range(start_year, end_year+1)

groups, total_mmb = grouping(start_year, end_year, cex_data_folder =␣
↪data_folder)
groups_mmb = {key: {} for key in groups.keys()}
for key in groups:
    for year in years:
        groups_mmb[key][year] = groups[key][year][['misparmb']]

group_analysis = {}
for key in groups.keys():
    group_number = list(groups.keys()).index(key) + 1
    total_groups = len(groups)
    print(f"Group {group_number}/{total_groups} ({key}) started.")
    combined_df, combined_secondary_df, combined_primary_df, yearly_price_index␣
↪= calculate_price_indexes(
        start_year, end_year, base_year, group_mmb=groups_mmb[key],␣
↪cex_data_folder=data_folder, verbose=False
    )
    group_analysis[key] = {
        'combined_secondary_df': combined_secondary_df,
        'combined_primary_df': combined_primary_df,
        'yearly_price_index': yearly_price_index
    }
```

```python
    print(f"Group {group_number}/{total_groups} ({key}) successfully computed.")
```

```
Group 1/11 (Arabs) started.

Loading price data: 100%|      | 4/4 [00:02<00:00,  1.46it/s]
Calculating price indexes: 100%|      | 4/4 [00:03<00:00,  1.12it/s]

Group 1/11 (Arabs) successfully computed.
Group 2/11 (Haredi) started.

Loading price data: 100%|      | 4/4 [00:02<00:00,  1.85it/s]
Calculating price indexes: 100%|      | 4/4 [00:03<00:00,  1.25it/s]

Group 2/11 (Haredi) successfully computed.
Group 3/11 (Low_inc) started.

Loading price data: 100%|      | 4/4 [00:02<00:00,  1.81it/s]
Calculating price indexes: 100%|      | 4/4 [00:02<00:00,  1.34it/s]

Group 3/11 (Low_inc) successfully computed.
Group 4/11 (High_inc) started.

Loading price data: 100%|      | 4/4 [00:02<00:00,  1.85it/s]
Calculating price indexes: 100%|      | 4/4 [00:05<00:00,  1.30s/it]

Group 4/11 (High_inc) successfully computed.
Group 5/11 (Young) started.

Loading price data: 100%|      | 4/4 [00:02<00:00,  1.67it/s]
Calculating price indexes: 100%|      | 4/4 [00:03<00:00,  1.02it/s]

Group 5/11 (Young) successfully computed.
Group 6/11 (Old) started.

Loading price data: 100%|      | 4/4 [00:02<00:00,  1.71it/s]
Calculating price indexes: 100%|      | 4/4 [00:04<00:00,  1.23s/it]

Group 6/11 (Old) successfully computed.
Group 7/11 (Low_SES) started.

Loading price data: 100%|      | 4/4 [00:02<00:00,  1.80it/s]
Calculating price indexes: 100%|      | 4/4 [00:04<00:00,  1.08s/it]

Group 7/11 (Low_SES) successfully computed.
Group 8/11 (High_SES) started.

Loading price data: 100%|      | 4/4 [00:02<00:00,  1.84it/s]
Calculating price indexes: 100%|      | 4/4 [00:02<00:00,  1.74it/s]

Group 8/11 (High_SES) successfully computed.
Group 9/11 (Muslim) started.

Loading price data: 100%|      | 4/4 [00:02<00:00,  1.78it/s]
Calculating price indexes: 100%|      | 4/4 [00:03<00:00,  1.25it/s]

Group 9/11 (Muslim) successfully computed.
Group 10/11 (Christian) started.
```

```
Loading price data: 100%|      | 4/4 [00:02<00:00,  1.83it/s]
Calculating price indexes: 100%|      | 4/4 [00:02<00:00,  1.86it/s]

Group 10/11 (Christian) successfully computed.
Group 11/11 (Druze) started.

Loading price data: 100%|      | 4/4 [00:02<00:00,  1.84it/s]
Calculating price indexes: 100%|      | 4/4 [00:01<00:00,  2.66it/s]

Group 11/11 (Druze) successfully computed.
```

[2]:

```python
gen_pop_df, gen_pop_secondary_df, gen_pop_primary_df,␣
↪gen_pop_yearly_price_index = calculate_price_indexes(start_year, end_year,␣
↪base_year, cex_data_folder=data_folder, verbose=False)
```

```
Loading price data: 100%|      | 8/8 [00:08<00:00,  1.00s/it]
Calculating price indexes: 100%|      | 8/8 [00:33<00:00,  4.14s/it]
```

[3]:

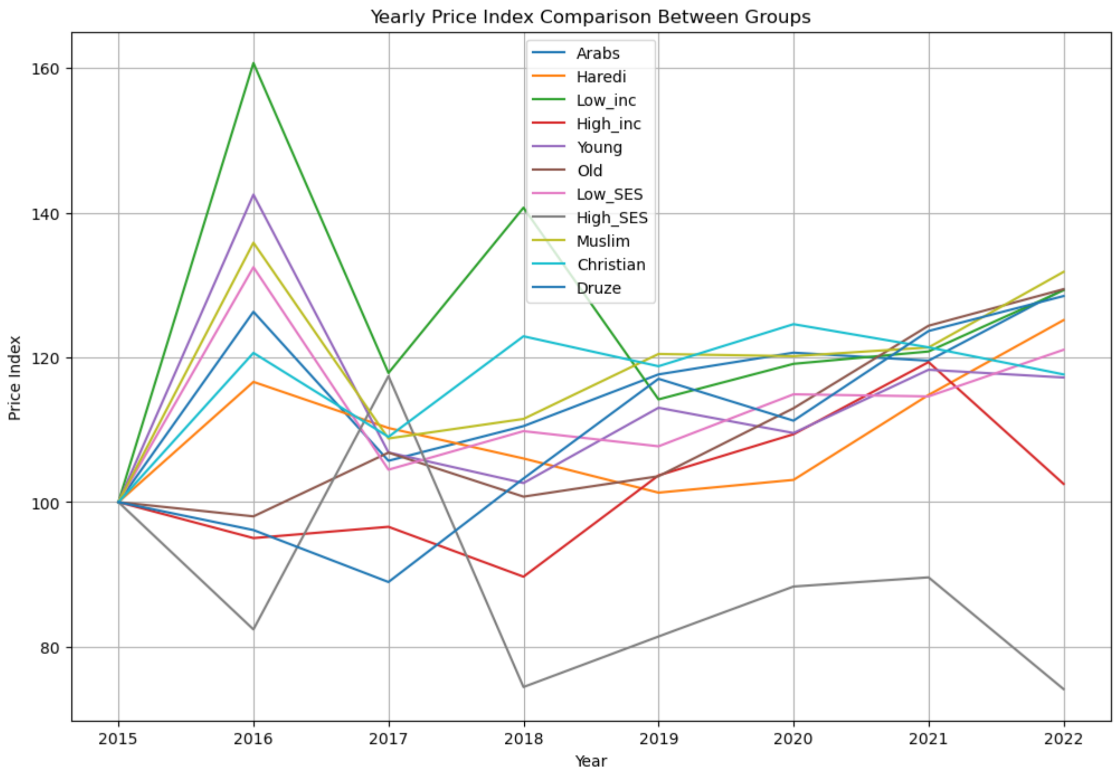
```python
group_counts = {group: {year: len(groups_mmb[group][year]) for year in␣
↪groups_mmb[group]} for group in groups_mmb}
# Create a dataframe with number of observations per year per group
observations_df = pd.DataFrame(group_counts).T

# Calculate the relative share of each group per year
total_observations_per_year = observations_df.sum(axis=0)
relative_share_df = observations_df.div(total_observations_per_year, axis=1) *␣
↪100

# Combine the absolute and relative values into a single dataframe
combined_df = observations_df.join(relative_share_df, rsuffix='_share')

# Display the dataframe
print(tabulate(combined_df, headers='keys', tablefmt='psql'))
```

```
+-----------+--------+--------+--------+--------+--------+--------+--------+----
----+--------------+--------------+--------------+--------------+--------------
+--------------+--------------+--------------+
|           |   2015 |   2016 |   2017 |   2018 |   2019 |   2020 |   2021 |
2022 |   2015_share |   2016_share |   2017_share |   2018_share |   2019_share
|   2020_share |   2021_share |   2022_share |
|-----------+--------+--------+--------+--------+--------+--------+--------+----
----+--------------+--------------+--------------+--------------+--------------
+--------------+--------------+--------------|
| Arabs     |   1136 |   1273 |   1327 |   1145 |   1103 |    513 |    951 |
727 |     11.5236  |     11.9351  |     12.2825  |     11.1295  |     10.9273  |
7.49014  |     11.1476  |      9.51944 |
| Haredi    |    734 |    778 |    757 |    786 |    565 |    440 |    551 |
595 |      7.44573 |      7.29421 |      7.00666 |      7.63997 |      5.59738 |
```

```
6.4243   |      6.4588   |      7.79102 |
| Low_inc    |    873 |    890 |    940 |    880 |    646 |    397 |    503 |
515 |      8.85575 |      8.34427  |      8.70048  |      8.55365 |      6.39984 |
5.79647  |      5.89614 |      6.74349 |
| High_inc   |   1667 |   1770 |   1741 |   1766 |   1848 |   1432 |   1479 |
1162 |      16.9101  |      16.5948   |      16.1144   |      17.1656   |      18.3079
|    20.9082   |      17.3368  |      15.2154  |
| Young      |   1340 |   1354 |   1358 |   1278 |   1108 |    718 |    877 |
820 |      13.593   |      12.6945   |      12.5694   |      12.4222  |      10.9768  |
10.4833   |      10.2802  |      10.7372  |
| Old        |   2091 |   2372 |   2348 |   2375 |   2279 |   1786 |   1779 |
1663 |      21.2112  |      22.2389   |      21.7327   |      23.0851  |      22.5778
|    26.0768   |      20.8534  |      21.7756  |
| Low_SES    |    616 |    695 |    728 |    654 |   1146 |    807 |   1204 |
1211 |       6.24873 |      6.51603  |      6.73825  |      6.35692 |      11.3533
|    11.7827   |      14.1132  |      15.857   |
| High_SES   |    121 |     95 |    102 |    127 |    189 |    163 |    153 |
146 |      1.22743 |      0.890681 |      0.944095 |      1.23445 |      1.8724  |
2.37991  |      1.79346 |      1.91175 |
| Muslim     |    880 |    999 |   1056 |    920 |    870 |    381 |    710 |
583 |      8.92676 |      9.36621  |      9.77416  |      8.94246 |      8.61898 |
5.56286  |      8.32259 |      7.63389 |
| Christian |    260 |    309 |    300 |    234 |    231 |    151 |    205 |
137 |      2.63745 |      2.89706  |      2.77675  |      2.27449 |      2.28849 |
2.2047   |      2.403   |      1.7939  |
| Druze      |    140 |    131 |    147 |    123 |    109 |     61 |    119 |
78 |      1.42017 |      1.2282   |      1.36061  |      1.19557 |      1.07985 |
0.890641 |      1.39491 |      1.02134 |
+-----------+--------+--------+--------+--------+--------+--------+--------+----
----+--------------+--------------+--------------+--------------+--------------
+--------------+--------------+--------------+
```

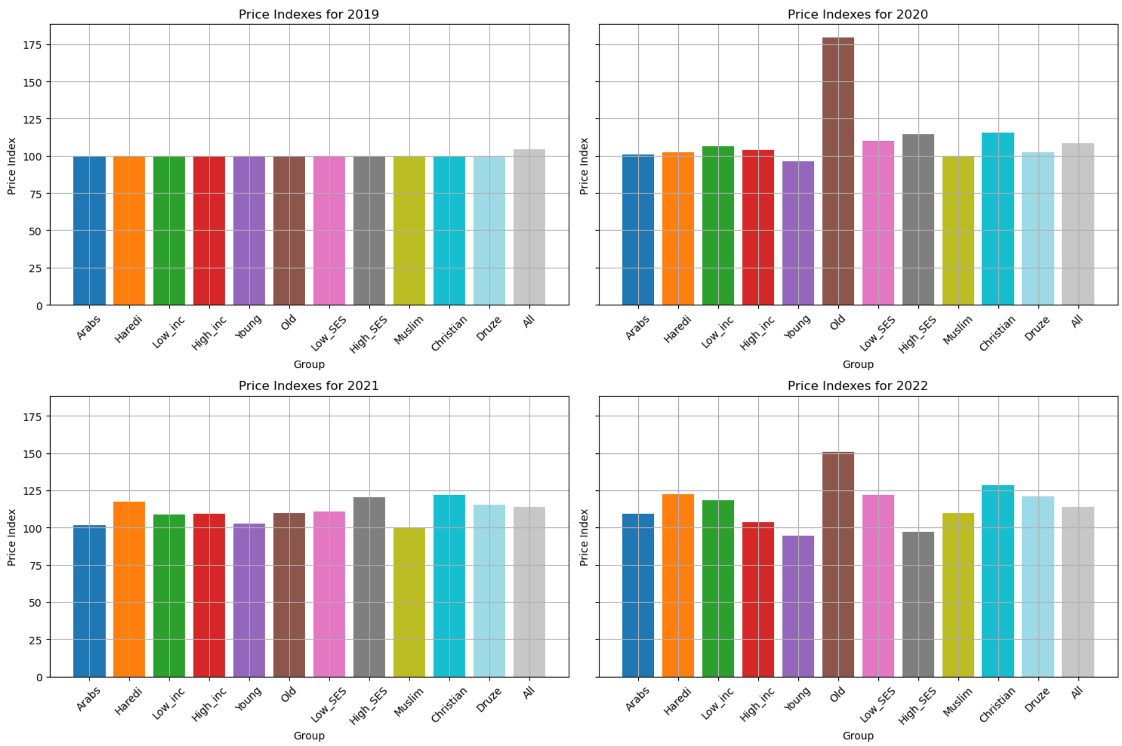
[4]:
```
# Extract yearly price indexes for each group
group_yearly_price_indexes = {group:␣
 ↪group_analysis[group]['yearly_price_index'] for group in group_analysis}

# Plot the yearly price indexes
plt.figure(figsize=(12, 8))
for group, price_indexes in group_yearly_price_indexes.items():
    years = list(price_indexes.keys())
    indexes = list(price_indexes.values())
    plt.plot(years, indexes, label=group)

plt.xlabel('Year')
plt.ylabel('Price Index')
plt.title('Yearly Price Index Comparison Between Groups')
plt.legend()
```

```
plt.grid(True)
plt.show()
```



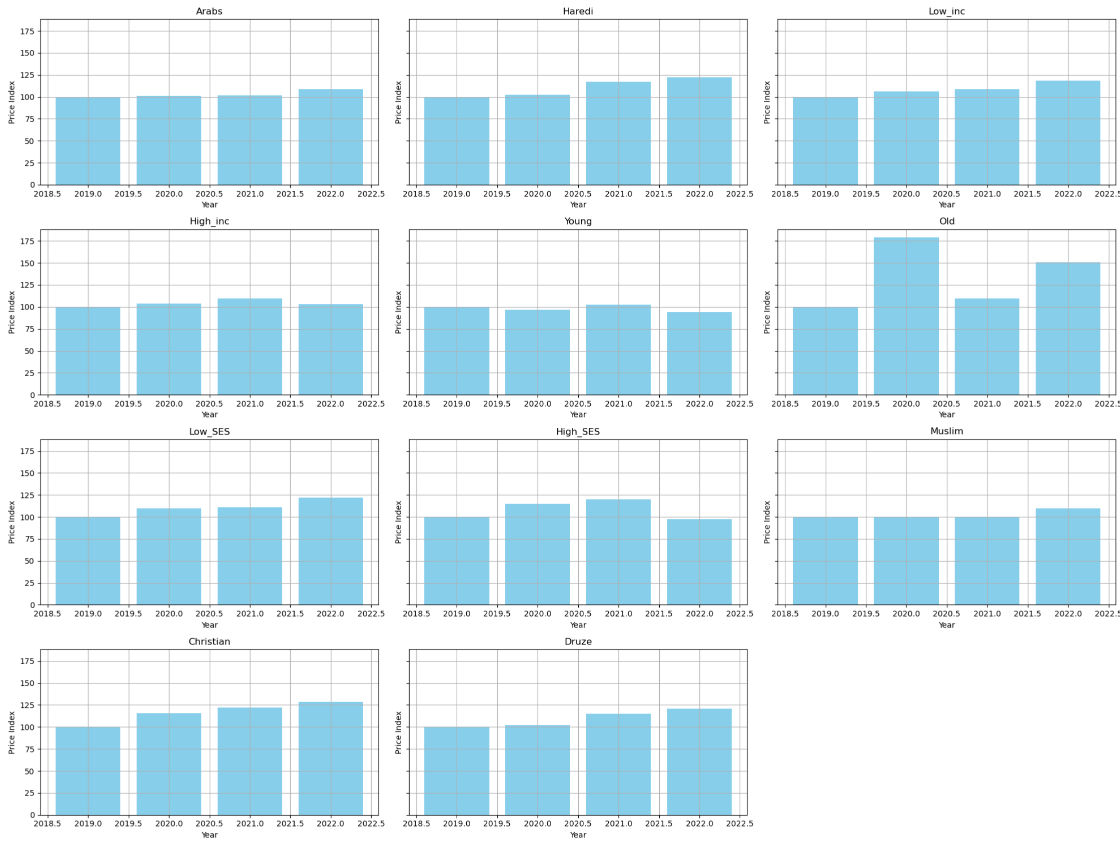
[7]:
```
# Extract yearly price indexes for each group including general population
group_yearly_price_indexes = {group:
↪group_analysis[group]['yearly_price_index'] for group in group_analysis}
group_yearly_price_indexes['All'] = gen_pop_yearly_price_index

# Define colors for each group
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b',
↪'#e377c2', '#7f7f7f', '#bcbd22', '#17becf', '#9edae5', '#c7c7c7']

# Create subplots for each year
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10), sharey=True)
axes = axes.flatten()

for i, year in enumerate(years[:len(axes)]):
    ax = axes[i]
    groups = list(group_yearly_price_indexes.keys())
    price_indexes = [group_yearly_price_indexes[group][year] for group in
↪groups]
    ax.bar(groups, price_indexes, color=colors)
```

```
    ax.set_title(f'Price Indexes for {year}')
    ax.set_xlabel('Group')
    ax.set_ylabel('Price Index')
    ax.grid(True)
    ax.tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```



[8]:
```
fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(20, 15), sharey=True)
axes = axes.flatten()

for i, (group, data) in enumerate(group_analysis.items()):
    yearly_price_index = data['yearly_price_index']
    years = list(yearly_price_index.keys())
    indexes = list(yearly_price_index.values())

    axes[i].bar(years, indexes, color='skyblue')
    axes[i].set_title(group)
    axes[i].set_xlabel('Year')
    axes[i].set_ylabel('Price Index')
    axes[i].grid(True)
```

```
# Remove any empty subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



[9]:
```
for group, analysis in group_analysis.items():
    # Calculate the weight differences between the group and the general␣
↪population for secondary categories
    weight_diff_df = analysis['combined_secondary_df'].copy()
    weight_diff_df['weight_diff'] = weight_diff_df['weight'] -␣
↪gen_pop_secondary_df['weight']

    # Sort by the absolute value of the weight differences in descending order
    weight_diff_df['abs_weight_diff'] = weight_diff_df['weight_diff'].abs()
    sorted_weight_diff_df = weight_diff_df.sort_values(by='abs_weight_diff',␣
↪ascending=False)

    # Display the top_n largest gaps using tabulate
```

```
    print(f"Group: {group}")
    print(tabulate(sorted_weight_diff_df.head(top_n)[['Year', 'prodcode',␣
↪'description', 'weight', 'weight_diff']], headers='keys', tablefmt='psql'))
    print("\n")
```

```
Group: Arabs
+-----+--------+------------+-----------------------------------------+---------
---+---------------+
|     |   Year |   prodcode | description                             |
weight |   weight_diff |
|-----+--------+------------+-----------------------------------------+---------
---+---------------|
| 249 |   2022 |        395 | Personal Jewelry And Watches            |
0.00278528 |    -0.131459  |
|  54 |   2019 |        384 | Other Transportation Expenses           |
0.0063836  |    -0.126337  |
| 116 |   2020 |        383 | Vehicle Expenses                        | 0.124914
|      0.11973   |
| 184 |   2021 |        393 | Personal Products And Cosmetics         |
0.0145025  |    -0.119572  |
| 179 |   2021 |        383 | Vehicle Expenses                        | 0.122662
|      0.115379  |
|  53 |   2019 |        383 | Vehicle Expenses                        | 0.125787
|      0.112121  |
| 242 |   2022 |        383 | Vehicle Expenses                        | 0.127554
|      0.106663  |
| 119 |   2020 |        391 | Cigarettes Tobacco And Smoking Supplies |
0.0300511  |    -0.103199  |
|  66 |   2020 |        302 | Meat And Poultry                        |
0.0924262  |     0.0855918 |
| 129 |   2021 |        302 | Meat And Poultry                        |
0.0886994  |     0.08524   |
+-----+--------+------------+-----------------------------------------+---------
---+---------------+


Group: Haredi
+-----+--------+------------+----------------------------------+------------+----
-----------+
|     |   Year |   prodcode | description                      |     weight |
weight_diff |
|-----+--------+------------+----------------------------------+------------+----
-----------|
| 184 |   2021 |        395 | Personal Jewelry And Watches     | 0.00416806 |
-0.129906  |
|  54 |   2019 |        384 | Other Transportation Expenses    | 0.00360779 |
-0.129113  |
| 119 |   2020 |        392 | Personal Services And Cosmetics  | 0.00463257 |
```

```
-0.128617  |
| 249 |   2022 |        397 | Organization Fees And Donations | 0.0118356  |
-0.122409  |
|  64 |   2020 |        300 | Bread Grains and Pastries       | 0.0956307  |
0.0921921 |
| 126 |   2021 |        300 | Bread Grains and Pastries       | 0.09406    |
0.0889357 |
| 188 |   2022 |        300 | Bread Grains and Pastries       | 0.091945   |
0.0812651 |
| 128 |   2021 |        302 | Meat And Poultry                | 0.0831521  |
0.0804045 |
|  66 |   2020 |        302 | Meat And Poultry                | 0.0845406  |
0.0777062 |
| 190 |   2022 |        302 | Meat And Poultry                | 0.0812823  |
0.0765053 |
+-----+--------+------------+---------------------------------+-----------+----
-----------+
```

```
Group: Low_inc
+-----+--------+------------+-----------------------------------+------------+--
-------------+
|     |   Year |   prodcode | description                       |     weight |
weight_diff |
|-----+--------+------------+-----------------------------------+------------+--
-------------|
| 119 |   2020 |        394 | Legal And Other Services          | 0.00321416 |
-0.130036  |
|  54 |   2019 |        385 | Mail Telephone And Communication  | 0.0110734  |
-0.121647  |
| 126 |   2021 |        302 | Meat And Poultry                  | 0.0938565  |
0.0887322 |
| 186 |   2022 |        302 | Meat And Poultry                  | 0.0931839  |
0.0810218 |
| 236 |   2022 |        383 | Vehicle Expenses                  | 0.0833307  |
0.080468  |
| 184 |   2022 |        300 | Bread Grains and Pastries         | 0.0631934  |
-0.0708808 |
|  52 |   2019 |        383 | Vehicle Expenses                  | 0.0838458  |
0.0703905 |
| 113 |   2020 |        383 | Vehicle Expenses                  | 0.0808254  |
0.0652736 |
| 173 |   2021 |        383 | Vehicle Expenses                  | 0.0794985  |
0.0648138 |
|  83 |   2020 |        332 | Electricity Gas And Fuel For Home | 0.062749   |
0.0626548 |
+-----+--------+------------+-----------------------------------+------------+--
-------------+
```

Group: High_inc

	Year	prodcode	description	weight	weight_diff
246	2022	383	Vehicle Expenses	0.158819	0.153794
249	2022	391	Cigarettes Tobacco And Smoking Supplies	0.00483538	-0.129409
184	2021	385	Mail Telephone And Communication	0.00645372	-0.127621
119	2020	384	Other Transportation Expenses	0.00634826	-0.126902
137	2021	308	Meals Outside Home	0.0956662	0.0893899
182	2021	383	Vehicle Expenses	0.0974391	0.0838352
118	2020	383	Vehicle Expenses	0.0971747	0.0834259
201	2022	308	Meals Outside Home	0.0877109	0.0769389
203	2022	311	Potatoes And Sweet Potatoes	0.00283584	-0.0441704
131	2021	302	Meat And Poultry	0.0481333	0.0412608

Group: Young

	Year	prodcode	description	weight	weight_diff
54	2019	384	Other Transportation Expenses	0.00469209	-0.128028
119	2020	385	Mail Telephone And Communication	0.0111502	-0.1221
249	2022	392	Personal Services And Cosmetics	0.0131016	-0.121143
184	2021	391	Cigarettes Tobacco And Smoking Supplies	0.0212876	-0.112787

181	2021	383	Vehicle Expenses	0.0935405	0.088328
53	2019	383	Vehicle Expenses	0.0979747	0.0843083
117	2020	383	Vehicle Expenses	0.0974486	0.0839201
245	2022	383	Vehicle Expenses	0.0902798	0.0701298
72	2020	308	Meals Outside Home	0.0739085	0.067667
136	2021	308	Meals Outside Home	0.0701319	0.0593855

Group: Old

	Year	prodcode	description	weight	weight_diff
184	2021	384	Other Transportation Expenses	0.00270334	-0.131371
119	2020	384	Other Transportation Expenses	0.00384282	-0.129407
249	2022	385	Mail Telephone And Communication	0.0092496	-0.124995
54	2019	382	Travel Abroad	0.00774532	-0.124975
55	2019	383	Vehicle Expenses	0.0974318	0.0920758
183	2021	383	Vehicle Expenses	0.0991997	0.0853741
247	2022	383	Vehicle Expenses	0.0983421	0.0847057
118	2020	383	Vehicle Expenses	0.0980098	0.084261
131	2021	302	Meat And Poultry	0.0684913	0.0616188
196	2022	302	Meat And Poultry	0.067964	0.061075

Group: Low_SES

	Year	prodcode	description	weight	weight_diff
249	2022	394	Legal And Other Services	0.0047145	-0.12953
184	2021	392	Personal Services And Cosmetics	0.00627667	-0.127798
119	2020	385	Mail Telephone And Communication	0.0147653	-0.118484
117	2020	383	Vehicle Expenses	0.104771	0.0912424
243	2022	383	Vehicle Expenses	0.105859	0.0908215
180	2021	383	Vehicle Expenses	0.10359	0.0834932
193	2022	302	Meat And Poultry	0.0859717	0.0832089
128	2021	300	Bread Grains and Pastries	0.0531601	0.0504126
191	2022	300	Bread Grains and Pastries	0.0534574	0.0483045
197	2022	306	Soft Drinks	0.0127775	-0.0423986

Group: High_SES

	Year	prodcode	description	weight	weight_diff
236	2022	383	Vehicle Expenses	0.230227	0.227364
53	2019	383	Vehicle Expenses	0.208694	0.195027
119	2020	396	Bags Suitcases And Other Products	0.00305701	-0.130193
54	2019	384	Other Transportation Expenses	0.00413969	-0.128581
184	2021	398	Expenses Not Elsewhere Specified	0.0120238	-0.12205
72	2020	308	Meals Outside Home	0.123332	

					0.11709
111	2020	383	Vehicle Expenses	0.103647	0.100235
193	2022	308	Meals Outside Home	0.0982297	0.0954668
130	2021	308	Meals Outside Home	0.117425	0.0714889
174	2021	383	Vehicle Expenses	0.0940957	0.0644457

Group: Muslim

	Year	prodcode	description	weight	weight_diff
249	2022	398	Expenses Not Elsewhere Specified	0.00274977	-0.131495
184	2021	396	Bags Suitcases And Other Products	0.00573652	-0.128338
54	2019	384	Other Transportation Expenses	0.00605455	-0.126666
176	2021	383	Vehicle Expenses	0.115603	0.112172
53	2019	383	Vehicle Expenses	0.119934	0.106268
119	2020	394	Legal And Other Services	0.0296476	-0.103602
113	2020	383	Vehicle Expenses	0.114868	0.0993161
239	2022	383	Vehicle Expenses	0.121383	0.0916623
66	2020	302	Meat And Poultry	0.0937284	0.086894
126	2021	302	Meat And Poultry	0.0883268	0.0832025

Group: Christian

	Year	prodcode	description

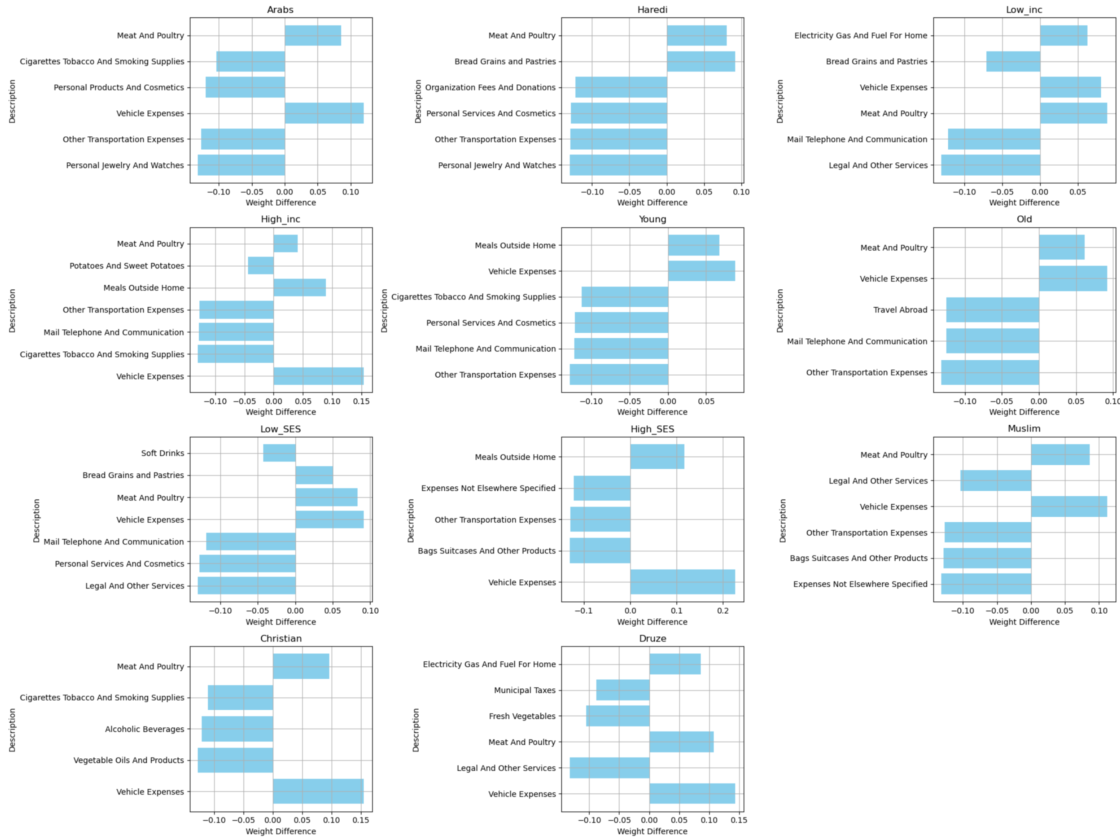
```
weight |   weight_diff |
|-----+--------+------------+------------------------------------------+---------
---+---------------|
| 225 |   2022 |        383 | Vehicle Expenses                         | 0.159651
|     0.154686  |
| 166 |   2021 |        383 | Vehicle Expenses                         | 0.150521
|     0.14903   |
|  51 |   2019 |        383 | Vehicle Expenses                         | 0.146481
|     0.141323  |
| 119 |   2021 |        301 | Vegetable Oils And Products              |
0.00548327 |    -0.127767  |
| 184 |   2022 |        307 | Alcoholic Beverages                      |
0.0122635  |    -0.121811  |
|  54 |   2019 |        391 | Cigarettes Tobacco And Smoking Supplies |
0.0221254  |    -0.110595  |
| 109 |   2020 |        383 | Vehicle Expenses                         | 0.132531
|     0.103045  |
|  64 |   2020 |        302 | Meat And Poultry                         |
0.0996246  |     0.096186  |
| 120 |   2021 |        302 | Meat And Poultry                         | 0.091822
|     0.0864336 |
| 179 |   2022 |        302 | Meat And Poultry                         |
0.0916972  |     0.0844143 |
+-----+--------+------------+------------------------------------------+---------
---+---------------+
```

```
Group: Druze
+-----+--------+------------+-----------------------------------+------------+-
--------------+
|     |   Year |   prodcode | description                       |     weight |
weight_diff |
|-----+--------+------------+-----------------------------------+------------+-
--------------|
| 207 |   2022 |        383 | Vehicle Expenses                  | 0.162452    |
0.143969  |
|  54 |   2019 |        394 | Legal And Other Services          | 0.000122185 |
-0.132598  |
|  98 |   2020 |        383 | Vehicle Expenses                  | 0.151386    |
0.129191  |
| 152 |   2021 |        383 | Vehicle Expenses                  | 0.152783    |
0.125564  |
|  48 |   2019 |        383 | Vehicle Expenses                  | 0.13748     |
0.122021  |
| 164 |   2022 |        302 | Meat And Poultry                  | 0.118103    |
0.107798  |
|  60 |   2020 |        302 | Meat And Poultry                  | 0.112965    |
0.107199  |
```

```
| 119 |   2021 |        312 | Fresh Vegetables                   | 0.0282493   |
-0.105001  |
| 184 |   2022 |        336 | Municipal Taxes                    | 0.0461439   |
-0.0879303 |
|  77 |   2020 |        332 | Electricity Gas And Fuel For Home | 0.104869    |
0.0865322 |
+-----+--------+------------+------------------------------------+-------------+-
--------------+
```

[10]:

```python
fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(20, 15), sharey=False)
axes = axes.flatten()

for i, (group, analysis) in enumerate(group_analysis.items()):
    # Calculate the weight differences between the group and the general␣
 ↪population for secondary categories
    weight_diff_df = analysis['combined_secondary_df'].copy()
    weight_diff_df['weight_diff'] = weight_diff_df['weight'] -␣
 ↪gen_pop_secondary_df['weight']

    # Sort by the absolute value of the weight differences in descending order
    weight_diff_df['abs_weight_diff'] = weight_diff_df['weight_diff'].abs()
    sorted_weight_diff_df = weight_diff_df.sort_values(by='abs_weight_diff',␣
 ↪ascending=False)

    # Replace NaN values in 'description' with an empty string
    sorted_weight_diff_df['description'] = sorted_weight_diff_df['description'].
 ↪fillna('')

    # Select the top n largest gaps
    top_n_weight_diff_df = sorted_weight_diff_df.head(top_n)

    # Plot the top n largest gaps
    axes[i].barh(top_n_weight_diff_df['description'],␣
 ↪top_n_weight_diff_df['weight_diff'], color='skyblue')
    axes[i].set_title(group)
    axes[i].set_xlabel('Weight Difference')
    axes[i].set_ylabel('Description')
    axes[i].grid(True)

# Remove any empty subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



[11]:
```python
fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(20, 15), sharey=False)
axes = axes.flatten()

for i, (group, analysis) in enumerate(group_analysis.items()):
    secondary_df = analysis['combined_secondary_df']
    # Filter for increases in price indexes
    increased_price_df = secondary_df[secondary_df['price_index'] > 100]
    # Calculate the contribution to the yearly price index
    increased_price_df = increased_price_df.copy()  # Ensure you're working with a copy
    increased_price_df.loc[:, 'contribution'] = increased_price_df['price_index'] * increased_price_df['weight']
    # Sort by contribution in descending order
    top_contributors = increased_price_df.sort_values(by='contribution', ascending=False).head(top_n)

    # Replace NaN values in 'description' with a placeholder text
    top_contributors['description'] = top_contributors['description'].fillna('No Description')
```

```
    # Plot the top contributors
    axes[i].barh(top_contributors['description'], top_contributors['contribution'], color='skyblue')
    axes[i].set_title(group)
    axes[i].set_xlabel('Contribution to Price Index')
    axes[i].set_ylabel('Description')
    axes[i].grid(True)

# Remove any empty subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```